# Problem 2 – Rainer

A Rainer is like a runner but in Rain. One who runs from the Rain. Donald is one good Rainer and he created a game where he dodges raindrops at lightning fast speed through some incomprehensible logic.

You will receive a sequence of integers – each of those integers, except the last one, form the game field.

You must take the last integer from that sequence – that is the initial index at which Donald steps.

The game goes so – you must decrease all of the integers in the sequence’ values by 1.

Then you must read an integer – the next index at which Donald steps.

You must repeat these steps until Donald gets wet.

If an integer reaches 0, that means a raindrop has fallen there. If Donald is on that position, he gets wet.

If an integer reaches 0, and Donald is not there, you must return the integer to its original value. (initial value)

When Donald gets wet, the program ends, and you must print the current sequence of integers, and the count of steps Donald has made (the initial index does not count as a step)

## Input

* On the first input line you will get the sequence of integers, separated by spaces.
* On the next several input lines you will be getting integers – the indexes.

## Output

* As output you must print the sequence of integers, separated by spaces, on one line.
* Then you must print the steps Donald has made on the second line.

## Constraints

* The count of the integers in the sequence will be [3, 100].
* The integers in the sequence will be in range [2, 100].
* The indexes that will be given to you will always be valid and inside the sequence.
* Allowed working time / memory: 100ms / 16MB.

## Examples

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 5 2 3 4 5 3  0  1  4  1  1 | 4 0 0 2 4  5 | Sequence – 5 2 3 4 5, Initial Index – 3  We decrease all by 1, Sequence – 4 1 2 3 4  We check if Donald is on an element 0. He is not, so we read next step. Index – 0. Steps – 1.  Sequence – 3 0 1 2 3. There is an element with value 0, but Donald is not there, we return it to its original value (2).  Sequence – 3 2 1 2 3. Index – 1. Steps – 2.  Sequence – 2 1 3 1 2. Index – 4. Steps – 3.  Sequence – 1 2 2 4 1. Index – 1. Steps – 4.  Sequence – 5 1 1 3 5. Index – 1. Steps – 5.  We decrease by 1, and it gets 4 0 0 2 4. Donald is on Index 1 – which is currently 0. He dies. No other steps are made, and the program ends. |
| 2 3 4 5 6 2  1  2  3  4  0 | 0 0 2 4 0  5 |  |